Informatik IIb

Objektorientierte Programmierung mit Java

Oliver Jack

Fachhochschule Jena
Fachbereich Elektrotechnik und Informationstechnik

Wintersemester 2010/11

Vorlesung 3. Klassen im Detail

Lernziele dieser Vorlesung

Methoden und Variablen

Zusammenfassung

Lernziele

- Kenntnis der Eigenschaften von Java-Klassen
- Kenntnis der Modifier und Sichtbarkeitsbereiche in Java
- Kenntnis von variablen Parameterlisten

Methoden einer Klasse

Signaturen einer Methode

- Methodenname, die Parameter und die Typen der Parameter sind die Signatur einer Methode.
- Rückgabewert gehört nicht dazu (im Gegensatz zu C++).
- Pro Klasse müssen alle Methoden eine verschiedene Signatur haben.

Beispiel

```
void main( String[] args );
String main( String[] argumete );
```

- Beide Methoden haben die selbe Signatur: (main, String[]).
- Nicht zusammen in einer Klasse deklarieren.

Modifier

Die Sichtbarkeit von Attributen und Methoden wird mit Hilfe folgender Modifier geregelt:

- public-Elemente sind in der Klasse selbst, in Methoden abgeleiteter Klassen und für den Aufrufer von Instanzen der Klasse sichtbar.
- protected-Elemente sind in der Klasse selbst und in Methoden abgeleiteter Klassen und in Klassen desselben Pakets sichtbar.
- private-Elemente sind lediglich in der Klasse selbst sichtbar.
- Elemente, die ohne einen der drei genannten Modifier deklariert wurden, werden als package scoped oder Elemente mit Standard-Sichtbarkeit bezeichnet. Sie sind nur innerhalb des Pakets sichtbar, zu dem diese Klasse gehört.

Modifier private

- private-Methoden oder -Attribute sind nur in der aktuellen Klasse sichtbar, in allen anderen Klassen bleiben sie dagegen unsichtbar.
- Dennoch können andere Objekte auf private Variablen eines Objekts zugreifen.
- Zugriff ist möglich für Objekte (Instanzen) der selben Klasse.

Modifier private

Beispiel für Zugriff auf private Attributen aus einem anderen Objekt der selben Klasse

```
public class privtest
{
  public static void main(String[] args)
  {
    ClassWithPrivateA a1 = new ClassWithPrivateA(7);
    ClassWithPrivateA a2 = new ClassWithPrivateA(11);
    a2.setOtherA(a1, 999);
    System.out.println("a1 = " + a1.toString());
    System.out.println("a2 = " + a2.toString());
  }
}
```

Modifier private

```
class ClassWithPrivateA
{
  private int a;
  public ClassWithPrivateA(int a)
  {
    this.a = a;
  }
  public void setOtherA(ClassWithPrivateA other,
                        int newvalue)
  {
    other.a = newvalue;
  }

  public String toString()
  {
    return "" + a;
  }
}
```

Modifier private

An der Ausgabe des Programms kann man erkennen, dass über das Objekt a2 auf private Attribute des Objekts a1 zugegriffen wurde:

```
a1 = 999
a2 = 11
```

Modifier protected

- Methoden oder Attribute vom Typ protected sind in der aktuellen Klasse sichtbar.
- Methoden oder Attribute vom Typ protected sind in abgeleiteten Klassen sichtbar.
- Darüber hinaus sind sie für Methoden anderer Klassen innerhalb desselben Pakets sichtbar.
- Sie sind jedoch nicht für Aufrufer der Klasse sichtbar, die in anderen Paketen definiert wurden.

Modifier public

- Attribute und Methoden vom Typ public sind im Rahmen ihrer Lebensdauer überall sichtbar.
- Nur Klassen, die als public deklariert wurden, sind außerhalb des Pakets sichtbar, in dem sie definiert wurden.
- In jeder Quelldatei darf nur eine Klasse mit dem Attribut public angelegt werden.

Standard (package scoped)

- Klassen, Methoden, Attribute mit Standard-Sichtbarkeit sind nur innerhalb des Pakets sichtbar, in dem sie definiert wurden.
- Sinnvoll, um in aufwendigeren Paketen allgemein zugängliche Hilfsklassen zu realisieren, die außerhalb des Pakets unsichtbar bleiben sollen.

Modifier static

- Attribute und Methoden mit dem Modifier static sind nicht an die Existenz eines konkreten Objekts gebunden, sondern existieren vom Laden der Klasse bis zum Beenden des Programms.
- Der static-Modifier beeinflusst bei Attributen ihre Lebensdauer.
- Der static-Modifier bei Methoden erlaubt den Aufruf, ohne dass der Aufrufer ein Objekt der Klasse besitzt, in der die Methode definiert wurde.
- Wird der Modifier static nicht verwendet, so sind Attribute innerhalb einer Klasse immer an eine konkrete Instanz gebunden.

Modifier final

- Attribute mit dem Modifier final dürfen nicht verändert werden, sind also Konstanten.
- Methoden mit dem Modifier final dürfen nicht überlagert werden.
- Klassen mit dem Modifier final dürfen nicht zur Ableitung neuer Klassen verwendet werden.
- Der final-Modifier kann auch auf Parameter von Methoden und lokale Variablen angewendet werden (ab JDK 1.1).
- Die Initialisierung muss dabei nicht unbedingt bei der Deklaration erfolgen, sondern kann auch später vorgenommen werden, aber nur einmal.

Modifier final

- Im Gegensatz zu C oder C++ gibt es bei als final deklarierten Objektparametern keine Möglichkeit, zwischen dem Objekt insgesamt und seinen einzelnen Elementen zu unterscheiden.
- Eine als final deklarierte Objektvariable wird zwar insgesamt vor Zuweisungen geschützt, der Wert einzelner Attribute kann jedoch verändert werden.
- Dies ist zu beachten bei Arrays, die in Java Objekte sind
- final bietet keinen Schutz gegen die unerwünschte Zuweisung eines Werts an ein einzelnes Element des Arrays.

Beispiel Konstante

```
public class Auto
{
  private static final double STEUERSATZ = 18.9;
}
```

Da Java keinen Präprozessor enthält und damit keine #define-Anweisung kennt, ist die beschriebene Methode das einzige Verfahren zur Deklaration von Konstanten in Java.

Beispiel Klassen-Methoden

- Klassen-Methoden existieren unabhängig von einer Instanz.
- Damit ist kein Zugriff auf Instanz-Variablen möglich.
- Einsatz, wo Funktionalitäten zur Verfügung gestellt werden, die nicht datenzentriert arbeiten oder auf primitiven Datentypen operieren, z. B. Klasse math.

```
public class Test_Math
{
  public static void main(String[] args)
  {
    double x, y;
    for (x = 0.0; x <= 10.0; x = x + 1.0) {
      y = Math.sqrt(x);
      System.out.println("sqrt("+x+") = "+y);
    }
  }
}
```

Variable Parameterlisten

- Möglich ab J2SE 5.0
- Der letzte Parameter einer Methode (nur dieser) kann nach dem Typbezeichner mit drei Punkten versehen werden.
- Damit kann beim Aufruf an dieser Stelle eine beliebige Anzahl Argumente des passenden Typs übergeben werden
- Der Compiler legt dann ein Array an.
- Beim Aufruf kann entweder das Array oder es können die einzelnen Elemente des Array angegeben werden.

```
public static void printArgs(String... args)
{
  for (int i = 0; i < args.length; ++i) {
    System.out.println(args[i]);
  }
}
```

Variable Parameterlisten

- Aufruf, beide Varianten sind gleichwertig

```
printArgs(new String[]{"so", "wird",
                       "es", "gemacht"});

printArgs("so", "wird", "es", "gemacht");
```

- Hier wird deutlich, warum nur der letzte Parameter variabel sein darf.
- Andernfalls könnte der Compiler unter Umständen nicht mehr unterscheiden, welcher aktuelle Parameter zu welchem formalen Parameter gehört.

Zusammenfassung

- Die Sichtbarkeit von Objekten wird in Java durch Modifier geregelt.
- Die Standardsichtbarkeit ist package-scoped.
- Gegenüber C++ gibt es den Modifier final, mit dem u. a. Konstanten deklariert werden.
- Wie in C gibt es in Java variable Paramterlisten für Methoden.